# 基于结构模块化实现 H. 264编码软件*)

张 剑 谭 劲

(华中科技大学计算机科学与技术学院 国家外存储系统专业实验室 武汉430074)

**摘 要** 本文详细分析了两大标准化组织最新提出的视频编码国际标准 H. 264中,如何采用模块化、结构化方法来实现 H. 264 编码软件,并对本实验室已完成的 TMZJ2004软件,进行了速度和性能方面的测试,给出了编码器率失真性能曲线图。

**关键词** 率失真,结构化,H. 264编码,比特流

## To Realize H. 264 Coding Software Based on the Structuring and Modularization

ZHANG Jian TAN Jin

(Huazhong University of Science and Technology, Wuhan 430074)

**Abstract** In this paper, how to realize H. 264 coding software through modularization and structuring has been analysed detailedly in the H. 264 video coding of international standard put forward newly by two organizations of standardization, and the speed and performance of software completed by our laboratory have been tested to obtain the curve drawing of Rate-Distortion of code device.

**Keywords** Rate-distortion, Structuring, H. 264 coding, Bit stream

ITU-T 的专家组在制定标准的过程中提供相应的测试模型(Test Model),如 H. 263的 TMN(Test Model Near-term)系列,H. 264的 TML(Test Model Long-term)系列和 JM(JVT Test Model)系列等[1]。这些测试模型通常采用标准 C 语言编写。为了验证编码器的正确性并对编码性能进行测试,许多编码软件的编写过程常常在测试模型上进行修改,结果导致编码器软件架构不合理,严重影响了软件的运行速度,并且不具备良好的可维护性、可移植性和代码重用性。本文着重阐述了如何采用模块化、结构化方法来实现 H. 264 编码软件,并分别针对不同的测试序列给出了编码器率失真性能曲线图,对测试性能结果进行了分析。

## 1 H. 264视频流语法

H. 264视频流语法,如图1所示。

语法元素的功能含义,如表1所示(见下页)。表中亮度块编码模式用4个字节的 CBPY 表示。如果一个8×8块中包含至少一个非零系数,则相应的字节位置1,否则该位为0表示该8×8块内没有系数传输。

色度块编码模式表示为,nc=0:没有色度系数;nc=1:存在非零的2×2色度块直流变换系数,所有的色度块交流系数为0,因此也不传输 AC 系数的 EOB 标志位;nc=2:可能存在非零的2×2色度块直流变换系数,同时存在至少一个非零的色度块交流系数。此时,需要传输2个 DC 系数的 EOB 标志位、2×4=8个4×4块的色度交流系数的 EOB 标志位。

## 2 模块与接口设计

结构化程序设计方法就是将程序模块看作是顺序结构、选择结构、重复结构和层次结构四种基本结构的集合[2]。它自顶向下进行设计,模块内的代码在执行结束之前不会调到其他模块,程序模块具有单入出口特征。在 H. 264编码语言中采用结构化设计方法,其程序的可维护性和可靠性都得到了明显提高,同时,模块结构化的设计方法也便于程序的正确性测试,提高了开发速度。如图2所示,给出了采用结构化思想设计的 H. 264编码软件的模块结构,各模块均采用标准 C 语言编写。图中 P1为编码器参数结构;P2为运动矢量信息结构。各模块的说明,如表2所示。这样,其编码软件又组成一个整体的模

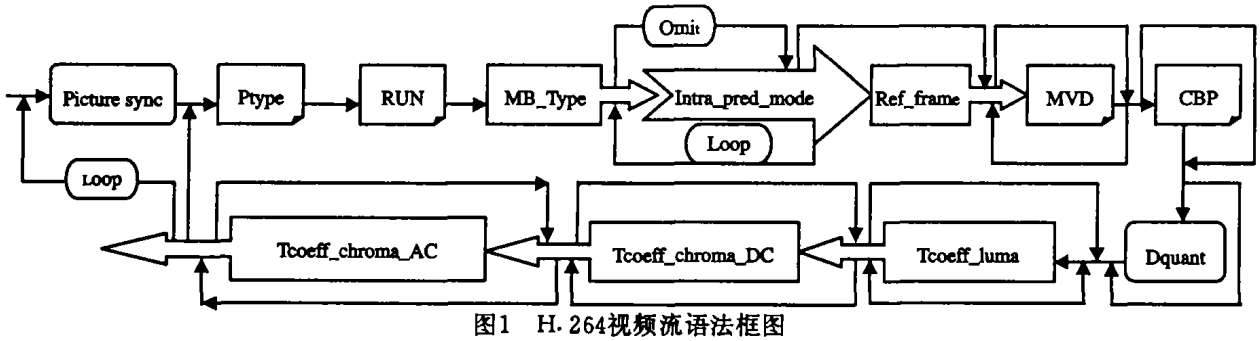


图1 H. 264视频流语法框图

*)本课题得到全国教育科学十五规划重点课题(No:AYA010034)和国家自然科学基金(No. 69735020)资助。张 剑 副教授,高级工程师,博士研究生,研究方向:计算机多媒体视频压缩算法及其实现。谭 劲 博士后,研究方向:多媒体数据存取与流媒体技术。

块，从面向对象的思想出发，这一整体的模块采用 C++进行封装，并为用户提供了方便的外部接口，便于大型应用系统其它模块的调用。

表1 语法元素的功能

| | | | |
|---|---|---|---|
| Picture Sync | 图像同步信息和头信息（共占用31个字节） | 时间参考 TR (Temporal Reference) | |
| | | 量化步长 PQP (Quantize step size) | |
| | | 图像类型 QCIF 或 CIF | |
| | | 结束标志 EOS (End Of Sequence) | |
| Ptype | 图像编码模式 picture Type | 0：只参考最近恢复图像的帧间预测模式 | |
| | | 1：多参考帧的帧间预测模式 | |
| | | 2：帧内模式 | |
| | | 3：B 帧编码模式 | |
| | | 4：多参考帧的 B 帧编码模式 | |
| Run | 表示宏块是否编码 | 如果该宏块被标注 skipped 则将前一帧解码的图像相应宏块复制到当前帧；Run 表示 P 或 B 图像中跳过的宏块个数 | |
| MB_Type | 宏块类型 | 对于帧内模式，宏块类型分为 | Intra_4×4 |
| | | | Intra_16×16 |
| | | 对于 P/SP 帧和 B 帧模式，宏块类型分为 | Inter |
| | | | Intra_4×4 |
| | | | tra16×16 |
| Intra_pred_mode | 帧内预测模式 | | |
| Ref_fram | 预测参考帧的位置 | 前一帧解码图像（即当前帧序号－1） | |
| | | 当前帧的前2帧（即当前帧序号－2） | |
| | | 当前帧的前3帧（即当前帧序号－3） | |
| MVD | 运动矢量 Motion Vector Data | 根据宏块的编码模式，传输相应的运动矢量数据 | |
| | | 对于不同编码块运动矢量的编码顺序和对高精度运动估值图像的插值编码方法 | |
| CBP | 编码块模式 Coded Block Patter | 表示8 × 8大小的亮度和色度块包含变换系数的参数 CBP：CBPY＋16×nc | |
| Dquant | 宏块级的量化步长范围是（－16，16） | 当没有非零变换系数传输时，Dquant 不存在 | |
| | | 当 CBP 表明宏块中有非零系数或对宏块采用16×16的帧内编码模式时，Dquant 才存在 | |
| QUANT new | 修改后的量化步长 | QUANT new＝modul032(QUANT01d＋Dquant＋32) | |
| Tcoeft_luma | 亮度传输系数 | | |
| Tcoeff_chroma_DC | 直流色度传输系数 | | |
| Tcoeff_chroma_AC | 交流色度传输系数 | | |

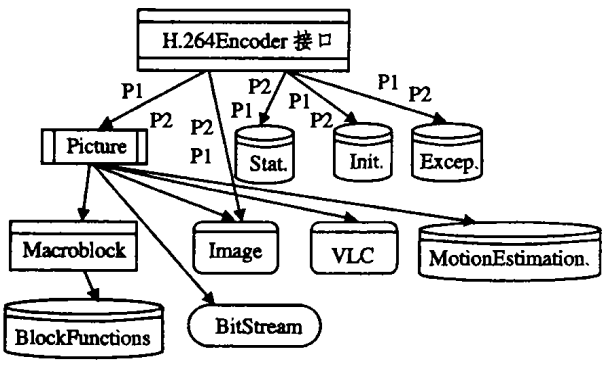


图2 H. 264 编码软件模块结构

通过封装，编码器的对外接口以两个 C++类的形式给出：H. 264Encoder 类负责实现编码功能，Enc_Option 类负责设置编码选项和控制参数，图3所示，为 H. 264Encoder 类的外部视图，更详细的内容请跟作者联系 zj3319896@163. net。

## 3 编码流程及控制

首先定义一个 H. 264Encoder 类的对象和一个空的 Enc_Option 结构，然后通过 Get Option 方法获取默认的选项并按照需要对其进行修改，最后依照修改后的选项调用 EncOneFrame 方法，进入编码循环，该编码中 H. 264Encoder 类实现流程图，如图4所示[1]。EncOneFrame 方法是有记忆的，每一次 EncOneFrame 的执行结果都与上一次 EncOne-

表2 模块的说明

| | |
|---|---|
| Picture 模块 | 负责完成一帧图像的编码，在接口上为其他模块提供一个 EncodeOneFrame 函数 |
| Stat 模块 | 负责计算编码后的信噪比等数据 |
| Init 模块 | 负责初始化编码器控制参数、比特流控制参数以及各种数据表；同时在接口上提供一个 GetSourceFrame 函数，负责获取待编码图像数据（通常是获取图像缓冲区的地址） |
| E×cep 模块 | 异常处理模块 |
| Macro block 模块 | 负责完成在各种编码模式下一个宏块的编码，在接口上提供一个 MB_Encode 函数 |
| Image 模块 | 负责完成图像处理相关如图像滤波、图像内插和图像扩展等功能 |
| VLC 模块 | 负责完成对各种语法元素如头信息，量化后的变换系数、运动信息等数据的变长编码 |
| Motion Estimation 模块 | 运动预测模块 |
| Block Functions 模块 | 负责完成各种以8×8块为单位的操作如运动补偿预测、差分、拷贝、变换等 |
| Bit Stream 模块 | 负责生成比特流 |

Frame 的执行情况有关，所以编码循环的控制策略由编码器的使用者决定。

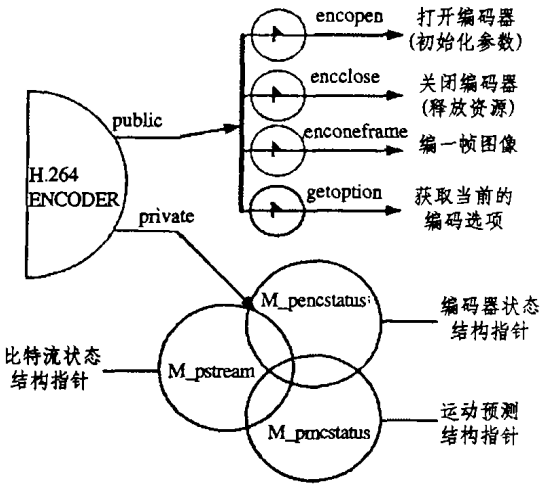


图3　H. 264Encoder 类的外部视图

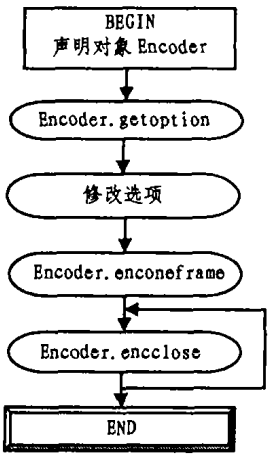


图4　Encoder 的编码流程图

EncOneFrame 方法的内部实现主要分为两部分，一部分为编码部分，另一部分为控制部分。编码部分主要由 Picture 模块提供的函数 EncodeOneFrame 来实现，其内部流程，如图5所示。其中，由 Macro block 模块提供的 MB_Encode 函数实现各种编码模式下宏块的编码过程。

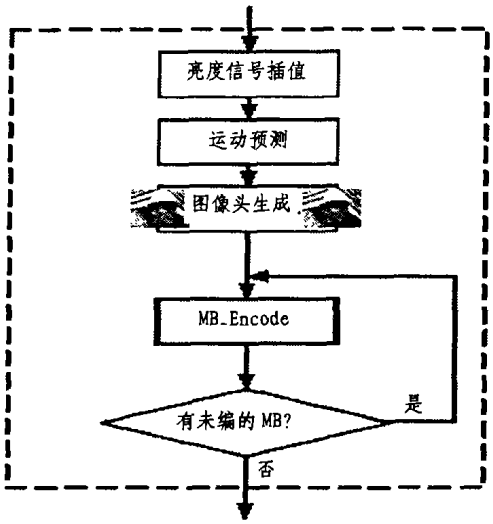


图5　EncOneFrame 的内部流程图

控制部分主要决定下一帧的编码类型。在 H. 264中，有三种编码图像类型：INTRA 类型，也称为帧内编码图像，这种图像禁止帧间编码模式；INTER 类型，也称为帧间编码图像，这种图像允许使用帧间和帧内编码模式；B 帧，也称为双向编码图像，这种图像除了允许使用帧内和普通的帧间编码模式外，还提供双向预测的模式。此外，为了进行编码控制，本文所描述的编码软件中还定义了两种特殊的模式，STORE 模式和 SKIP 模式。INTRA 帧的解码过程不需要参考图像，所以，定期地采用 INTRA 帧可以防止变换不匹配和传输错误造成的噪声积累。STORE 模式则是为了编 B 帧而设计的，由于 B 帧采用双向预测，其编码顺序与采集顺序并不一致，在编某一个 B 帧时，当前采集到的图像需要预先存储下来，作为下一个 B 帧的输入图像，若设两个 INTER 帧间有两个 B 帧，则 B 帧的编码过程示意，如表3所示。

表3　B 帧的编码过程

| 帧序号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | —— | N | —— |
|---|---|---|---|---|---|---|---|---|---|---|
| 帧类型 | INTER | STORE | STORE | INTER | B | B | INTER | —— | INTER | —— |
| 编码器操作 | 编帧0 | 存帧1不编码 | 存帧2不编码 | 编帧3 | 存帧4不编码 | 存帧5不编码 | 编帧6 | —— | 编帧 N | —— |

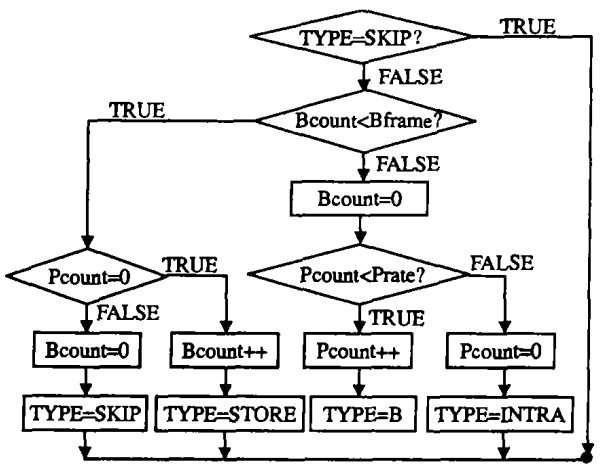


图6　控制部分流程图

由于最初的两个应当编作 B 帧的图像没有输入，只需要把当前采集到的图像存储下来，因而记作 STORE 类型。对于 STORE 类型的编码图像，编码器没有码字输出。SKIP 模式则是为进行简单的速率控制而设计的，在 SKIP 模式下，编码器不做任何操作。

图6所示为控制部分的流程图，其中有两个变量和两个控制参数。两个变量为 BFrame 和 PRate，分别表示两个 INTER 帧之间的 B 帧数目和两个 INTRA 帧之间的 INTER 帧数目，它们由编码器的使用者进行设置。两个控制参数为 Bcount 和 Pcount，它们在编码器初始化时被设为零。

## 4　比特流控制

在 H. 264/AVC 的比特流中，比特流的生成主要依赖于 Bit Stream 模块提供的函数 put bits，为了适应不同的传输环境，它支持两种比特输出的方式。以帧为单位的比特输出方式，一帧图像编码完成后输出生成的全部比特；以字节为单位的比特输出方式，当生成的比特达到给定字节数时即将其输出。然而，需要注意的是，输出比特流的具体实现过程是由用

户依据传输环境来完成的，如宏块的传输顺序由宏块分配映射(Macroblock Allocation Map)确定。H.264/AVC 支持柔性宏块排序方式(Flexible Macroblock Ordering)，宏块可以组成一个或几个片组，每一个片组单独传输，当一个片组发生丢失时，可以利用与之临近的已经正确接收到的另一片组中的宏块进行有效的错误掩盖，以 TCP 或 UDP 的方式发送，存为文件或送到串口等。因此，其它模块在调用编码器之前必须为编码器提供一个回调函数，由其完成具体的比特流输出过程。这里，回调函数是引自 Windows 系统下用户界面设计的一个概念，即由用户编写而被系统调用的函数，也即其它模块设计而被当前模块调用的函数。这个回调函数供有5个输入参数，分别为比特缓冲区地址、字节数、时间信息、图像类型和一个附加信息区的指针。Bit Stream 模块内部通过调用这个函数来完成具体的比特流输出过程。这种设计方法有利于提高编码器的通用性，使其内部不做任何修改便能应用于各种传输环境，这也是结构化程序设计的思想[3]。

## 5 H.264的 TMZJ2004仿真程序

H.264的 TMZJ2004编码程序的设计流程，主要分为以下几部分：编码参数设置、视频数据输入、帧内编码、帧间编码、熵编码、码流输出。本文只解释编码主程序，其它程序本文不再详细讨论。

### 5.1 编码参数设置

编码参数的设置通过对特定的配置文件(如 encoder.cfg)进行操作来完成[4]。为方便对参数的存取，采用结构化存储方式，即参数名称，类型，数值。

参数设置程序 encoder.Cfg 包括文件操作参数#Plles、编码控制参数#Encoder Control、定义输入参数结构体 Input ammeters、Mapping。

读取配置文件中的参数通过 configure()函数调用以下四个函数实现：static char * Get Config File Content(char * Filename)、static void Parse Content(char * buf,intbufsize)、staticint Parameter Name To MapInde × (char * S)、static void PatchInp()。

Configure()函数，首先在 Get Config File Content()函数内存中为各个参数分配空间，然后打开配置文件 encoder.cfg，将参数读入内存；然后，通过 Parse Content()函数解析每个参数的具体含义，将相应参数赋值，最后通过 PatchInp()函数检验参数的一致性。

### 5.2 编码主程序

为简化程序，增加程序可读性，采用多个子函数嵌套编程方法。在主程序编写中，主要进行编码参数获取和配置，初始化各种图像编码参数，分配和释放内存，编码和统计编码结果等操作。编码主程序具体编码过程在 image.c 中的 encode_one_flame 函数完成编码一个 I 帧、P 帧或 B 帧。encode_one_flame 函数中又根据编码模式依次调用 image.c 中的 encode_one_shce 函数编码片、macroblock.c 中的 encode_one_macroblock 函数编码宏块等，按照编码顺序分层进行编码。

编码主程序中用到了三个主要结构体：Input Parameters、Image Parameters、Stat Parameters。Input Parameters 结构体包含了所有输入配置变量；Image Parameters 结构体定义了当前帧的所有编码属性；Stat Parameters 结构体定义了所有统计结果属性。

## 6 实验测试分析

使用 TMN 和 H.264Encoder、TML 测试模型，如表4所示，给出编码器性能和速度的测试结果。

表4 编码器速度测试结果(QP=5，128M/PentiumIII450MHz)

| 序列 | 编码器 | 帧数目 | 编码时间(秒) | 编码速率(秒) |
|---|---|---|---|---|
| Container_qcif | TMN | 300 | 40 | 7.21 |
| | H.264encoder | 300 | 27 | 11.01 |
| Carphone.qcif | TMN | 380 | 52 | 1.01 |
| | H.264encoder | 380 | 38 | 9.98 |

实验结果表明，H.264Encoder 类的编码速度要比测试模型 TMN 的编码速度快40%，这主要是由于 H.264Encoder 类在内部实现的过程中，采用了缓冲区指针传递代替内存数据拷贝，在一定程度上节约了编码时间。

图7和图8分别针对不同的测试序列给出了编码器的性能测试结果，从图中所示的率失真性能曲线可以看出，与 TMN 相比，H.264Encoder 类的编码性能没有明显降低[5]。

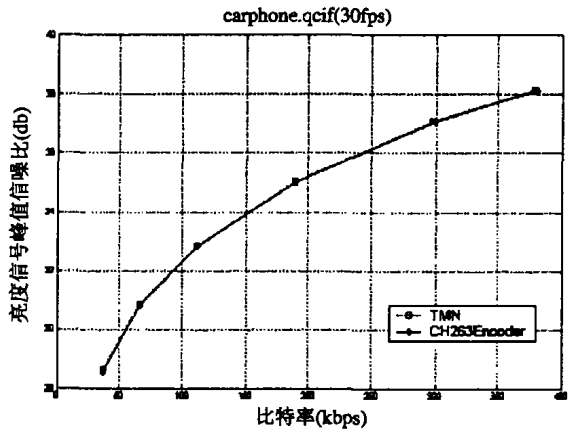


图7 编码器性能测试结果(1)

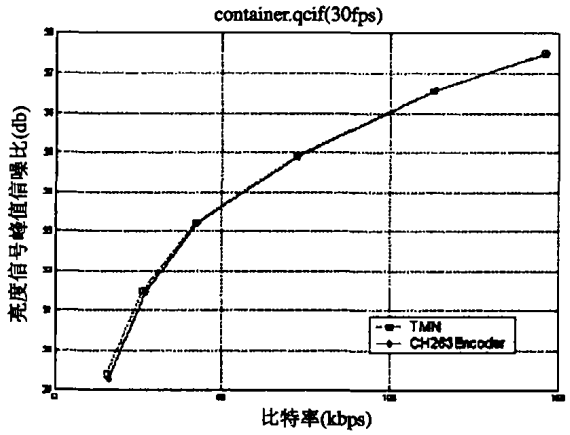


图8 编码器性能测试结果(2)

## 参考文献


1 Wiegand T. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG 7th Meeting: Pattaya, Thailand, March 2003
2 Hallapuro A, et al. Lo× complexity transform and quantization. JVT of ISO/IEC MPEG and ITU-T VCEG, Docs. JVT-B038 and JVT-B09, Jan. 2002
3 Wiegand T. Joint Video Team (JVT) of ISO/IECMPEG and ITU-T VCEG. Geneva, Switzerland, January 29-February 1, 2002
4 陈兵旗. Visual C++6.0实用图像处理. 清华大学出版社，2004
5 张剑. H.264标准的量化16位算法. 计算机科学，2004(11)